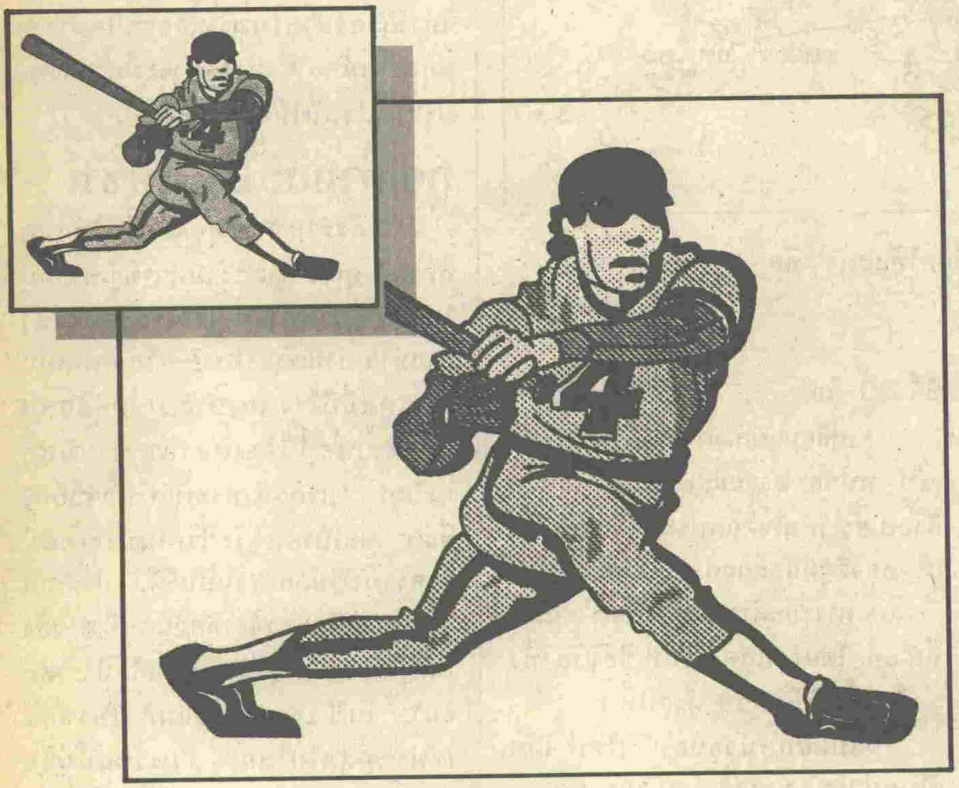
ขยายภาพด้วยปาสคาล (ตอนที่ 1)

อาจหาญ สัตยารักษ์



"เทคนิคการดัดแปลงภาพกราฟิกให้ดูแปลกตาขึ้น อย่างเช่น วิธีการขยายภาพ การกลับภาพซ้ายขวา การทำภาพหัวกลับ และการทำภาพกลับขาวดำ"

การสร้างภาพกราฟิกเป็นเรื่องสนุก เพราะว่าเราสามารถจะเขียนภาพเพียงภาพเดียวแล้วนำภาพที่ได้นี้ไปแสดงผลในรูปแบบต่าง ๆ ได้ ทั้งยังสามารถนำไปขยายให้ได้ภาพโตขึ้นกว่าเดิมกี่เท่าก็ได้ วิธีการสร้างภาพบนจอภาพนั้น ผู้เขียนได้กล่าวถึงมาหลายครั้งแล้ว เช่น ในบทความในฉบับที่ 50 (หน้า 267-273) ซึ่งคิดว่าผู้อ่านคงจะได้สร้างรูปเก็บไว้กันบ้างแล้ว

คราวนี้เราจะมาพูดถึงรูปแบบของการนำภาพที่สร้างไว้แล้วนั้น มาแสดงผลให้ได้หน้าตาแปลก ๆ ออกไปจากเดิม กล่าวคือ จะนำมาทำเป็นภาพกลับขาวดำ, ภาพกลับซ้ายขวา, ภาพหัวกลับ, ภาพขยาย เป็นต้น

ผู้เขียนขอพูดถึงเรื่องนี้เป็น 2 ตอน คือ

ตอนที่ 1 เป็นการขยายภาพให้แสดงบนจอภาพ การจะพิมพ์ภาพลงเครื่องพิมพ์ก็ใช้วิธีลอกจอภาพลงไปโดยตรงกันเลย (ใช้โปรแกรม WaitTogo ของผู้เขียนในไมโครฯฉบับที่ 44 หน้า 195-198 โดยอย่าลืมเพิ่มต้นโปรแกรมเรียกใช้ยูนิต dos ด้วย) ส่วนจะขยายได้โตสุดก็เท่าที่ขนาดของจอภาพจะยอมให้เท่านั้น ซึ่งเป็นเรื่องที่กล่าวถึงในฉบับนี้ทั้งหมด

ตอนที่ 2 เป็นการขยายภาพด้วยความสามารถของเครื่องพิมพ์ โดยจะใช้วิธีอ่านข้อมูลจากหน้าจอภาพ (โดยอ่านจากภาพขนาดเล็กสุดที่กล่าวถึงในครั้งนี้) ส่งไปให้เครื่องพิมพ์ทำงานซึ่งจะสามารถขยายภาพให้โตขึ้นได้มากกว่าที่กล่าวถึงในฉบับนี้

โปรแกรมทั้งหมดในฉบับนี้ ผู้เขียนได้จัดทำเป็นซอฟต์เมล์ด้วย ท่านที่ไม่ประสงค์จะนั่งคีย์โปรแกรมก็ติดต่อขอใช้บริการได้เช่นเคย

อ่านภาพจากไฟล์ Slib.shp

ผู้อ่านคงเคยใช้แผ่นดิสค์โปรแกรม Print Master(PM) มาแล้ว จะเห็นว่าภาพที่ปรากฏในโปรแกรม Slib.shp นี้มีจำนวนถึง 122 ภาพ แต่ละภาพมีความสวยงามมากและอยากได้ภาพเหล่านั้นมาใช้ในโปรแกรมปาสคาลของเรากันบ้าง คราวนี้คงสมใจท่านแล้ว เพราะโปรแกรมในรูปที่ 1 ต่อไปนี้จะช่วยให้ท่านเรียกภาพทั้งหมด 122 ภาพนั้นมาใช้ร่วมกับภาพอื่น ๆ ที่เราเขียนได้

ขอให้ดูโปรแกรมในรูปที่ 1 กันก่อน

โปรแกรมนี้มีชุดคำสั่งอยู่ 3 ชุดคำสั่ง คือ

1. opengraph; เพื่อเข้ากราฟิกโหมด

2. LoadPic; เพื่ออ่านข้อมูลจากไฟล์ภาพชื่อ Slib.shp ในที่นี้ผู้เขียนอ่านมาถึงภาพลำดับที่ 110 เท่านั้น เพราะภาพข้างหลังเป็นภาพลายเส้นเป็นส่วนใหญ่ (ภาพมี 122 ภาพ โดยอยู่ในข้อมูลลำดับที่ 0-121) ดังนั้นภาพลำดับที่ 110 ก็คือภาพที่ 111 ตามการนับของคนปกตินั่นเอง ตามโปรแกรมผู้เขียนให้อ่านภาพนี้จากช่องอ่านแผ่นบันทึกช่อง B ในคำสั่ง assign(f, 'b:Slib.shp'); หากท่านใช้ช่องอ่านแผ่นช่องอื่นก็ให้แก้ไขคำสั่งให้ตรงด้วย

ชุดคำสั่งนี้จะสั่งเปิดไฟล์ภาพเมื่อค่า no=0 และทำการปิดแฟ้มเมื่อค่า no=110

```
{  Readpic.pas   ReadPic From Print Master  By Arthan Sattayarak }

USES crt,dos,printer,graph;
type pic=array[1..577] of byte;
     PicRec=record
            PicL:pic;
            end;
var ch:char;
    F,Fsave:file of PicRec;
    picLoad:pic;
    PicR:Picrec;      stt:string;
    ScanLine,x,xx:integer;
    n1,n2:byte;       k,Size,EndScan:word;

PROCEDURE OpenGraph;
var
     GrDriver,GrMode :integer;
begin
   GrDriver :=0; GrMode:=2;
   InitGraph(GrDriver, GrMode, '');
end;

PROCEDURE LoadPic(no:byte);
begin
 if no=0 then begin
   assign(f,'b:Slib.shp');
   Reset(f);  end;
   seek(f,no);     read(f,PicR);
   with PicR do PicLoad:=picL;
 if no=110 then  close(f);
end;

PROCEDURE ShowPic(x,y:integer; No:byte);
var k,xst:word;
begin
 LoadPic(no);  xst:=x;
 for k:=1 to 572 do
  begin
       setlinestyle(4,Picload[k+4],1);
       line(x-7,y,x+8,y);      Inc(x,8);
    if k mod 11 =0 then begin  x:=xst; Inc(y) end;
  end;
end;
```

รูปที่ 1 โปรแกรมชุดคำสั่งอ่านภาพจากไฟล์ Slib.shp ของ Print Master

ภาพในไฟล์นี้แต่ละภาพจะใช้ข้อมูล 577 ไบต์ มีตัวแปรคือ PicL,PicLoad เป็นตัวแปรรองรับข้อมูลที่อ่านมาได้ โดยอ่านมาครั้งละหนึ่งภาพ ถ้าใครจะเขียนอ่านมาทั้งหมดประมาณ 70 กิโลไบต์ ก็ต้องใช้คำสั่งให้เหมาะสมด้วย เพราะจะไปเกี่ยวข้องกับหน่วยความจำ (ผู้เขียนเห็นว่าอย่างไรเราก็คงใช้ทีละภาพก็เลยอ่านมาครั้งละหนึ่งภาพก็เพียงพอแล้ว) โดยข้อมูลของหนึ่งภาพนี้ก็ได้จัดเรียงลำดับกันไว้ โดยมีส่วนหัวและส่วนท้ายเป็นรหัสเปิดและปิดให้รู้การจบภาพหนึ่งภาพ

3. ShowPic; ชุดคำสั่งเพื่ออ่านข้อมูลของแต่ละภาพนั้นให้แสดงเป็นภาพบนจอภาพ ให้ผู้อ่านสังเกตว่าในโปรแกรมนั้นใช้เพียงข้อมูลลำดับที่ 5 ถึง 576 (k+4) เท่านั้น ไฟล์นี้เก็บข้อมูลเฉพาะส่วนที่เป็นภาพด้วยข้อมูลขนาด 11×52 ซึ่งเมื่อคูณกันจะได้ค่าเป็น 572 (576-5+1=572)

ข้อมูลที่อ่านได้จึงจัดลงมาเป็นภาพได้ 52 แถว ๆ ละ 88 จุด (11×8 = 88) ตามชุดคำสั่งก็เพิ่มค่า y ทุก ๆ ครั้งที่ลงจุดได้ครบ 11 ไบต์(ไบต์ละ 8 จุด) จากการตรวจค่า k mod 11 = 0 นั่นเอง

ขอให้สังเกตการลงจุดในชุดคำสั่งนี้ซึ่งผู้เขียนใช้วิธีลงจุดครั้งละ 8 จุดเลย โดยอาศัยคำสั่งเกี่ยวกับรูปแบบการกำหนดเส้น (SetLineStyle) เข้ามาช่วยและใช้รูปแบบที่กำหนดได้เอง คือ UserBitln (ค่า = 4) ในบรรทัดที่ 1 ถัดจากคำสั่ง Begin ดังนี้ setLineStyle(4, PicLoad[k+4],1); ซึ่งลงจุดด้วยคำสั่งลากเส้น คือ Line(x-7,y,x+8,y);

ชุดคำสั่งในโปรแกรมนี้จะนำไปใช้ร่วมกับโปรแกรมหลักในรูปที่ 2, โปรแกรมในรูปที่ 3 และใช้กับโปรแกรมในรูปที่ 5 ด้วย โดยผู้อ่านอาจจะใช้วิธีเรียกมาด้วย^KRหรือใช้คำสั่ง $include ก็ได้

ส่วนโปรแกรมหลักในรูปที่ 2 ก็คือส่วนที่จะเรียกโปรแกรมในรูปที่ 1

```
{======================= Readpic.pas ======================}
begin
 opengraph;
 for k:=0 to 110 do
  begin
    setfillstyle(1,0); Bar(33,2,420,90);  setfillstyle(1,1);
    Showpic(32,2,k);
    str(k,stt); outtextxy(40,70,'pic no.'+stt);
    ch:=readkey;   if ch=#27 then halt(0);
  end;
 closegraph;
end.
```

รูปที่ 2 โปรแกรมหลักเพื่ออ่านภาพจาก โปรแกรมในรูปที่ 1 แสดงทีละ 1 ภาพ

เข้ามาทำงานและให้ปรากฏภาพบนจอตามลำดับจากภาพที่ 0 ไปจนถึงภาพที่ 110 (อยากดูถึงภาพสุดท้าย ก็ให้เปลี่ยนค่า 110 ไปเป็น 121) ถ้าอยากจบก่อนถึงภาพสุดท้ายก็ให้กดแป้น ESC ภาพที่ได้จะสวยงามเพียงใดก็ขอให้รันดูบนจอภาพกันเอง เพราะผู้เขียนไม่ได้พิมพ์ภาพจากโปรแกรมมาให้ดู

ในซอฟต์เมล์นั้นผู้เขียนได้รวมโปรแกรมในรูปที่ 1 และรูปที่ 2 เข้าด้วยกันในชื่อไฟล์ ReadPic.pas (โดยมีไฟล์ Slib.shp มาให้ด้วย)

ทำภาพกลับขาวดำ กลับซ้ายขวาและภาพหัวกลับ

เราก็ได้ดูภาพจากไฟล์รูปภาพของ Print Master มาจนลานตากันไปหมดแล้ว

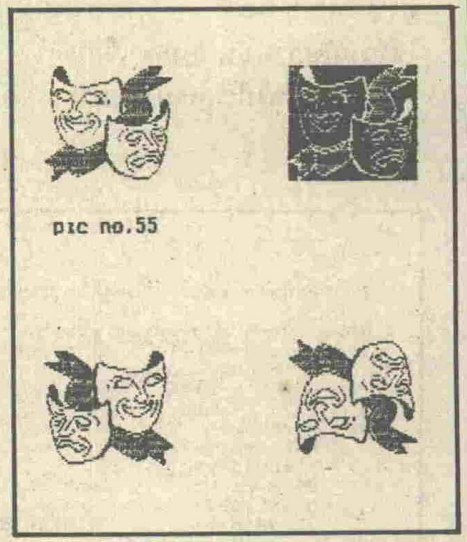
```
PROCEDURE InversePic(x,y:integer);
begin
     getimage(33,2,120,53,Picload); PutImage(x,y,Picload,4);
end;

PROCEDURE LeftToRightPic(x,y:integer);
var xx,yy,Xpix:integer;
begin
    for xx:=33 to 120 do
        begin
          getimage(xx,2,xx,53,Picload);
          putimage(x,y,Picload,0);
          dec(x);
        end;
 end;

 PROCEDURE HeadReversePic(x,y:integer);
 var xx,yy,Xpix:integer;
 begin setcolor(1);
      for yy:=2 to 53 do
        begin
          getimage(33,yy,120,yy,Picload);
          putimage(x,y,Picload,0);
          dec(y);
        end;
end;

{======================= PicDisp.pas ======================}
begin
 opengraph;
 for k:=0 to 110 do
  begin
    setfillstyle(1,0); Bar(0,0,719,347);  setfillstyle(1,1);
    Showpic(32,2,k);
    str(k,stt); outtextxy(40,70,'pic no.'+stt);
    InversePic(200,2);
    LeftToRightPic(120,130);
    HeadReversePic(200,176);
    ch:=readkey;   if ch=#27 then halt(0);
  end;
 closegraph;
end.
```

รูปที่ 3 โปรแกรมเพื่อกลับภาพขาวดำ, กลับซ้ายขวา, และทำภาพหัวกลับ

ต่อไปนี้เราจะนำภาพที่ได้มาทำให้เกิดภาพในลักษณะอื่น ๆ กันต่อไป โดยในที่นี้ขอเสนอภาพใหม่ 3 แบบ คือ ภาพกลับขาวเป็นดำหรือดำเป็นขาว ภาพกลับซ้ายเป็นขวา และภาพหัวกลับ โดยขอเสนอวิธีอ่านภาพจากจอภาพมาแทนจากการอ่านจากข้อมูลของตัวแปร PicLoad

โดยการใช้คำสั่งง่าย ๆ คือ GetImage และ PutImage เข้ามาช่วยจัดการก็สามารถทำได้ทันที

โปรแกรมในรูปที่ 3 จะเป็นส่วนของการสร้างภาพ 3 แบบนี้ ขอให้ลองศึกษาเองก่อนที่จะอ่านคำอธิบาย



รูปที่ 4 ภาพที่ได้จากโปรแกรมในรูปที่ 3 (เป็นภาพลำดับที่ 55 ของไฟล์ Slib.shp)

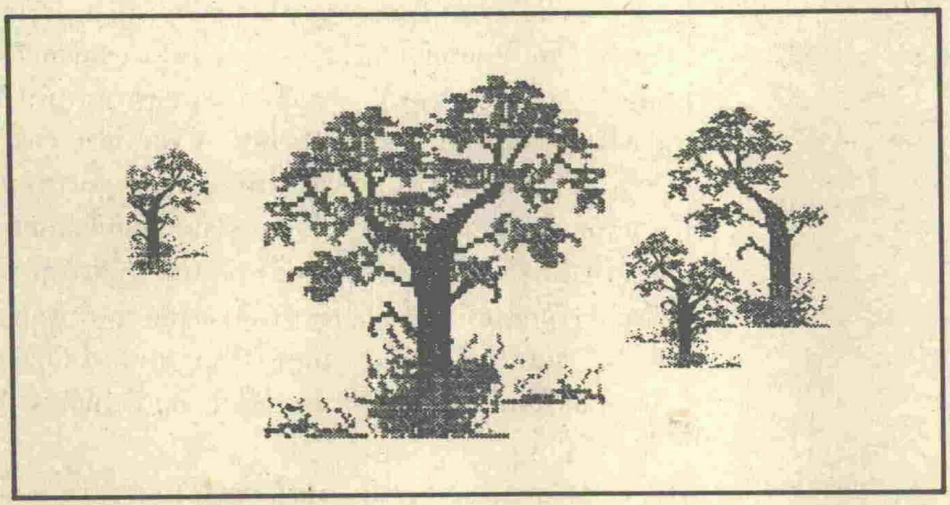
ชุดคำสั่งแรกคือ InversePic ชุดคำสั่งนี้ใช้วิธีจำภาพทั้งหมดใหม่อีกครั้ง ใช้ตัวแปรเดิมคือตัวแปร PicLoad ซึ่งเดิมใช้อ่านข้อมูลจากไฟล์ แต่คราวนี้มาใช้คำสั่ง GetImage(33,2,120,53, Picload); ขอให้สังเกตว่าตัวแปร PicLoad เป็นข้อมูลชนิดไบต์ ไม่ใช่ข้อมูลชนิดพอยน์เตอร์(Pointer) ดังนั้นจึงไม่ต้องใช้ ^ ตามหลังตัวแปร Picload หรือไม่ใช่ PicLoad^ ขอบเขต 33,2, 120,53 ก็คือตำแหน่งของภาพที่แสดงบนจอภาพไว้เดิมนั้นเอง แล้วนำมาแสดงภาพกลับขาวดำด้วยคำสั่ง PutImage

ต้นฉบับไม่มีหน้า 301-302

No This Page 301-302 In Original

รูปที่ 7 (ต่อ)

```
VAR    ch:char;    xpix:integer; k:word;

PROCEDURE OpenGraph;
var
   GrDriver,GrMode :integer;
begin
   GrDriver :=0; GrMode:=2;
   InitGraph(GrDriver, GrMode, '');
end;

PROCEDURE Drawpic(x,y:integer; pic:string; Mode:byte);
var k,Xstart,i :integer;
 begin
  Xstart:=x;
   for k:=1 to length(pic) do
    begin
      if pic[k]='x' then begin   setfillstyle(9,1);
         bar(xstart,y,Xstart+xpix,y+xpix);  end;
      if pic[k]='z' then begin   setfillstyle(1,1);
         bar(xstart,y,Xstart+xpix,y+xpix);  end;
      if pic[k]='c' then begin   setfillstyle(6,1);
         bar(xstart,y,Xstart+xpix,y+xpix);  end;
      if mode=0 then Inc(Xstart,xpix+1) Else Dec(Xstart,xpix+1);
   end;          setfillstyle(1,1);
end; {Drawpic }

PROCEDURE Pic(xx,yy:integer; Size,Mode:byte);
var no :integer;
    pic :string[120];
begin
    Xpix:=size-1;
  for no:= 1 to 66 do
    begin
      pic:=pix[no];
      Drawpic(xx,yy,pic,Mode);
      Inc(yy,Xpix+1);
    end;
end;  {Pic }

PROCEDURE Draw_Pic;
BEGIN
   Pic(630,20,2,1); Pic(0,45,1,0);    Pic(45,55,1,1);
   Pic(520,95,1,1); Pic(478,110,1,0);
   Pic(210,0,3,0);  Pic(340,20,3,1);
end;

{======================= DrawTree ==========================}
begin
   opengraph;
    Draw_Pic;
    Ch:=readkey;
 CloseGraph;
end.
```



รูปที่ 8 ภาพที่ได้จากการสร้างภาพจากลายปักแสดงการขยายภาพ,การกลับซ้าย ขวา

โดยการกำหนดรูปลายปักลงในตัวแปร Pix (ตามโปรแกรมเป็นขนาด 66 ข้อความ) และใช้อักษร z, x, c เพียง 3 ตัวที่บอกความแตกต่างของรูป 4 แบบ (รวมช่องว่างอีก 1 ด้วย โดยจากการพิมพ์ผู้เขียนได้จัดเครื่องพิมพ์ให้พิมพ์บรรทัดชิดกันเพื่อจะได้เปรียบเทียบกับภาพบนจอได้ง่ายขึ้น)

ชุดคำสั่ง DrawPic จะวาดภาพที่อ่านได้ในรูปแบบที่กำหนดด้วยคำสั่ง SetFillStyle 3 แบบ เมื่อตรวจพบว่าเป็น z, x หรือ c โดยส่งตัวแปร Mode ให้ไปเพื่อจะได้ภาพเหมือนต้นแบบ (mode = 0) หรือจะได้ภาพกลับซ้ายขวา (เมื่อ mode = 1) ส่วนการขยายภาพนั้นเป็นการกำหนดจากตัวแปรกลาง Xpix อีกทีหนึ่ง โดยที่ชุดคำสั่งนี้เป็นการสร้างจากข้อความ pic ครั้งละหนึ่งแถวเท่านั้น

ชุดคำสั่ง Pic เป็นชุดคำสั่งที่จะวาดภาพทั้งหมดลงจอภาพ โดยมีตัวแปร Size เป็นตัวแปรกำหนดว่าจะให้ได้ภาพขนาดโตเท่าใด (นำไปเปลี่ยนค่าตัวแปร Xpix อีกที คือ Xpix: = Size-1;)

ชุดคำสั่ง Draw_Pic ก็จะทำให้เกิดภาพต้นไม้ทั้งหมดดังที่พิมพ์มาให้ดูในรูปสุดท้าย คือรูปที่ 8 ซึ่งโดยการสั่งให้กลับซ้ายขวาและขยายภาพได้ จึงได้ต้นไม้ที่มีรูปร่างต่างกันได้เพียงแต่การจัดให้ส่วนของลำต้นมาวาดทับที่เดียวกันหรือใกล้ ๆ กัน

หวังว่าผู้อ่านคงได้แนวคิดของการวาดภาพ, ขยายภาพ เพื่อการพัฒนาโปรแกรมด้านกราฟิกของท่าน ฉบับต่อไปคือตอนที่ 2 จะได้พูดถึงการขยายภาพโดยอาศัยเครื่องพิมพ์ซึ่งจะได้ภาพขนาดโตมากกว่าวิธีขยายโดยอาศัยจอภาพ

พบกันใหม่..ในฉบับหน้า